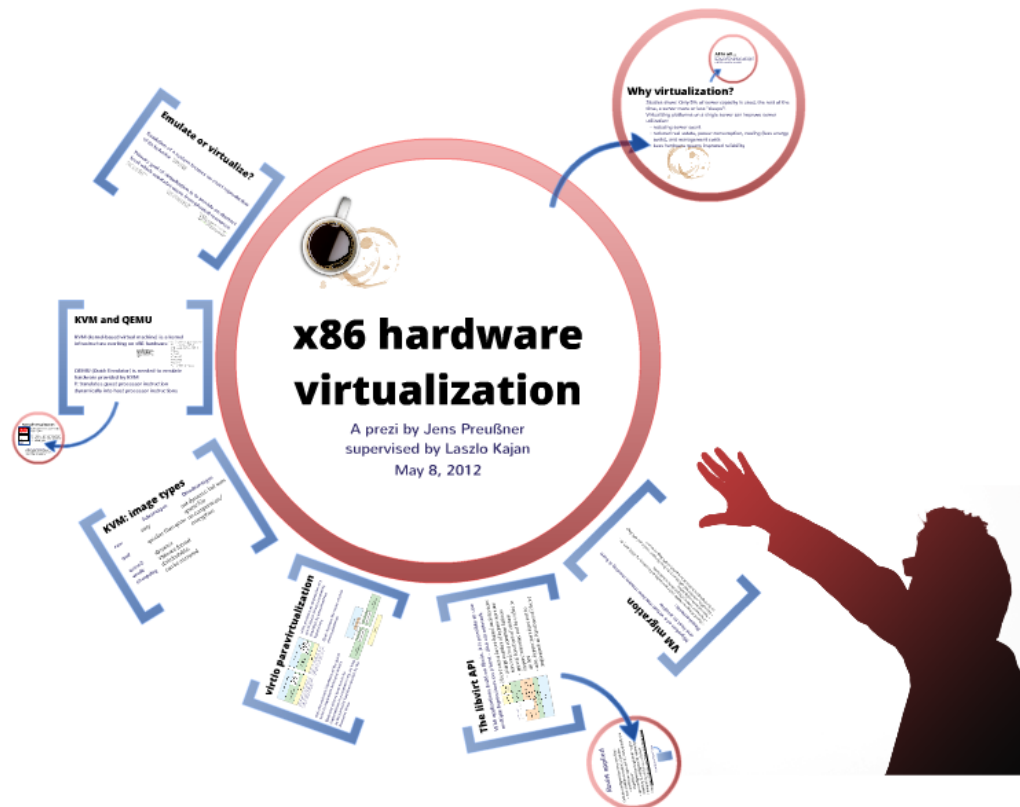


Thank you for your attention!



x86 hardware virtualization

A prezi by Jens Preußner
supervised by Laszlo Kajan
May 8, 2012



...or virtualize?
...ulation of a system focuses on exact reproduction
of its behavior
...ary goal of virtualization is to provide an abstract
... which separates users from physical resources

...MU
...achine) is a kernel
...6 hardware
...List of supported guest systems:
• Linux (32 and 64 bit)
• Windows (32 and 64 bit)
• Solaris
• AIX
• FreeBSD
• Solaris
• other BSD-Derivates
...needed to emulate
...instruction
...sor instructions

...age types
Advantages
Disadvantages
not dynamic, but uses
sparse file
no compression/
encryption
quicker than qcow
dynamic
VMware format
distributable,
can be mirrored

...ns moving it from
... have to be from the same vendor, and CPU flags
... both hosts
... paths and locations, e.g. iSCSI, NFS, etc.
...st be supported, e.g. CPU flags on source

costs), and management costs
Less hardware means improved reliability

Emulate or virtualize?

Emulation of a system focuses on exact reproduction of its behavior

This means, that its function is reproduced, the functionality achieving this may be different!

Primary goal of virtualization is to provide an abstract level which separates users from physical resources

This enables us to share hardware platforms, operating systems, storage devices, or network resources among many users

But it is also possible to merge heterogeneous devices into a homogeneous environment and make users "believe" that they use their hardware exclusively

Name conventions:

- actual machine, where virtualization takes place is called **host**
- a virtual machine is called **guest**
- software that creates a virtual machine on the host hardware is called a **hypervisor** or Virtual Machine Monitor (like KVM)

system t

This means, that its function is reproduced, the functionality achieving this may be different!

Emulate or virtualize?

Emulation of a system focuses on exact reproduction of its behavior

This means, that its function is reproduced, the functionality achieving this may be different!

Primary goal of virtualization is to provide an abstract level which separates users from physical resources

This enables us to share hardware platforms, operating systems, storage devices, or network resources among many users

But it is also possible to merge heterogeneous devices into a homogeneous environment and make users "believe" that they use their hardware exclusively

Name conventions:

- actual machine, where virtualization takes place is called **host**
- a virtual machine is called **guest**
- software that creates a virtual machine on the host hardware is called a **hypervisor** or Virtual Machine Monitor (like KVM)

level which

This enables us to share hardware platforms, operating systems, storage devices, or network resources among many users

Virtualization is to p es users from p

But it is also possible to merge heterogeneous devices into a homogeneous environment and make users "believe" that they use their hardware exclusively

Provide an abstract physical resources

Name conventions:

- actual machine, where virtualization takes place is called **host**
- a virtual machine is called **guest**
- software that creates a virtual machine on the host hardware is called a **hypervisor** or Virtual Machine Monitor (like KVM)

KVM and QEMU

KVM (kernel-based virtual machine) is a kernel infrastructure working on x86 hardware


x86 describes a processor architecture, which provides hardware virtualization techniques like Intel VT and AMD-V

List of supported guest systems:

- Linux (32 and 64 Bit)
- Windows (32 and 64 Bit)
- Haiku
- AROS
- ReactOS
- FreeDOS
- Solaris
- other BSD-Derivates

QEMU (Quick Emulator) is needed to emulate hardware provided by KVM

It translates guest processor instruction dynamically into host processor instructions



x86 describes a processor architecture, which provides hardware virtualization techniques like Intel VT and AMD-V

KVM and QEMU

KVM (kernel-based virtual machine) is a kernel infrastructure working on x86 hardware

x86 describes a processor architecture, which provides hardware virtualization techniques like Intel VT and AMD-V

List of supported guest systems:

- Linux (32 and 64 Bit)
- Windows (32 and 64 Bit)
- Haiku
- AROS
- ReactOS
- FreeDOS
- Solaris
- other BSD-Derivates

QEMU (Quick Emulator) is needed to emulate hardware provided by KVM

It translates guest processor instruction dynamically into host processor instructions

KVM and QEMU

KVM (kernel-based virtual machine) is a kernel infrastructure working on x86 hardware

x86 describes a processor architecture, which provides hardware virtualization techniques like Intel VT and AMD-V

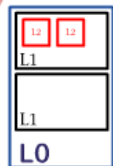
List of supported guest systems:

- Linux (32 and 64 Bit)
- Windows (32 and 64 Bit)
- Haiku
- AROS
- ReactOS
- FreeDOS
- Solaris
- other BSD-Derivates

QEMU (Quick Emulator) is needed to emulate hardware provided by KVM

It translates guest processor instruction dynamically into host processor instructions

Nested virtualization



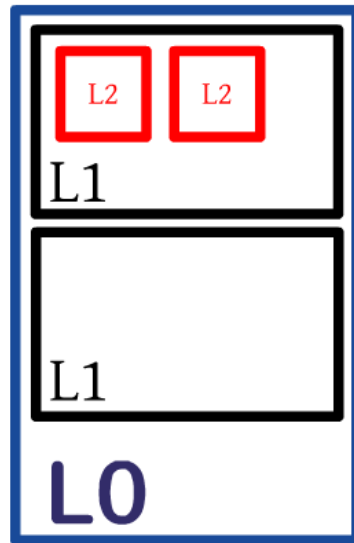
enables you to run a guest inside a regular guest

L0 = physical host, host hypervisor
L1 = regular guest, guest hypervisor
L2 = nested guest

Why nested virtualization?

- Development of hypervisor software
- Using multiple guests on a providers host, which is shared with other users

Nested virtualization



enables you to run a guest inside a regular guest

L0 = physical host, host hypervisor

L1 = regular guest, guest hypervisor

L2 = nested guest

Why nested virtualization?

- Development of hypervisor software
- Using multiple guests on a providers host, which is shared with other users

KVM: image types

Advantages

Disadvantages

raw

easy

not dynamic, but uses
sparse file

qed

quicker than qcow

no compression/
encryption

qcow2

dynamic

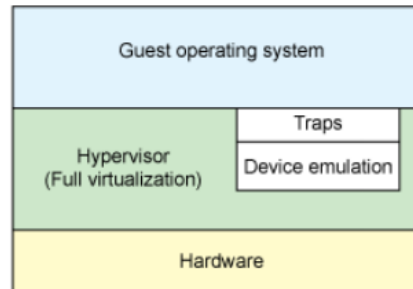
vmdk

VMware format

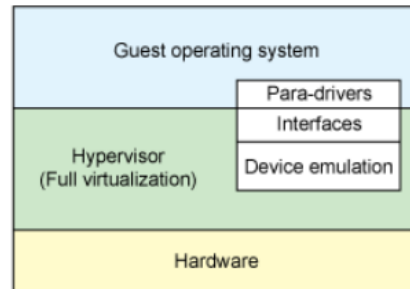
sheepdog

distributable,
can be mirrored

virtio paravirtualization



In full virtualization, the guest operating system runs on top of a host hypervisor. The guest is unaware that it is being virtualized and requires no changes to work in this configuration.



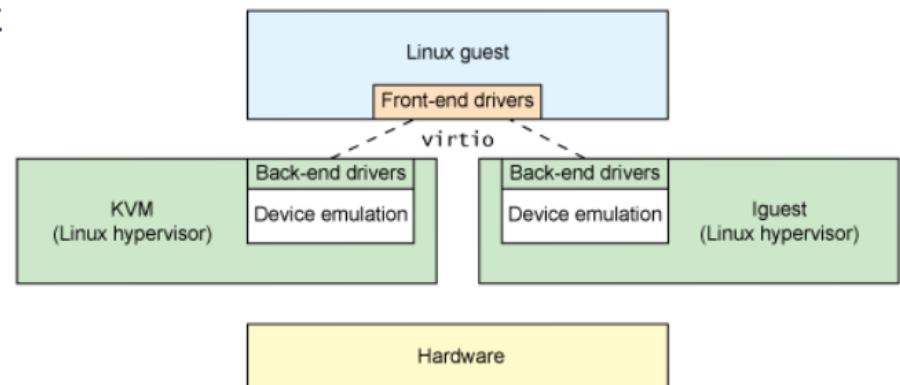
Conversely, in paravirtualization, the guest operating system is not only aware that it is running on a hypervisor but includes code to make guest-to-hypervisor transitions more efficient

virtio provides an abstraction of a set of device drivers commonly emulated by paravirtualized hypervisors

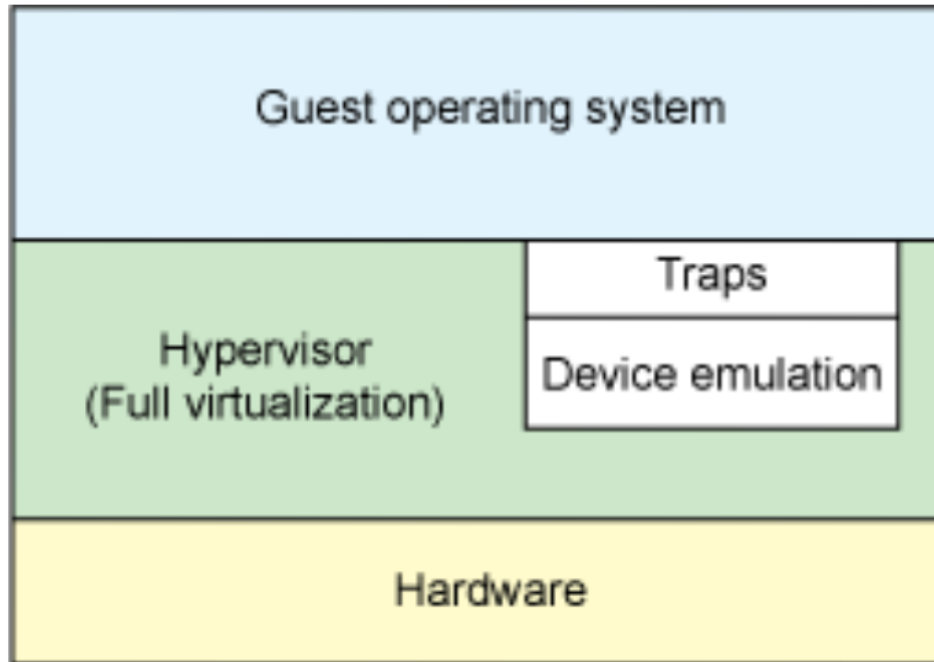
Goal: increase the reuse of code across platforms

disk and network interfaces of the guest need to implement front-end drivers

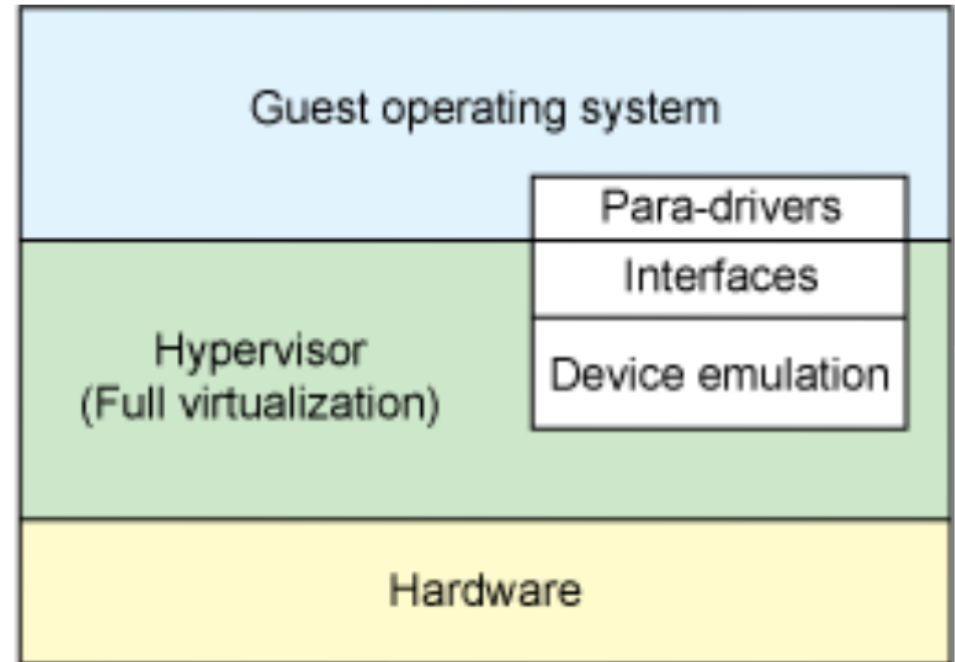
Back-end drivers have not to be implemented in a common way, as long as they provide functions needed by the front-end driver



virtio paravirt

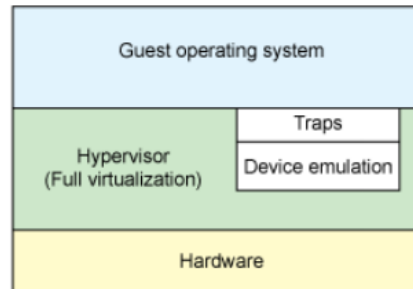


In full virtualization, the guest operating system runs on top of a host hypervisor. The guest is unaware that it is being virtualized and requires no changes to work in this configuration.

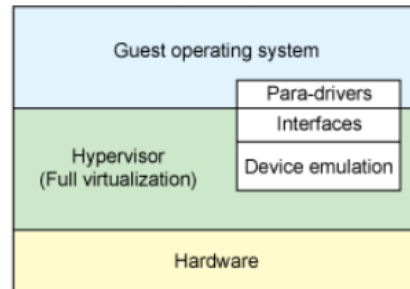


Conversely, in paravirtualization, the guest operating system is not only aware that it is running on a hypervisor but includes code to make guest-to-hypervisor transitions more efficient

virtio paravirtualization



In full virtualization, the guest operating system runs on top of a host hypervisor. The guest is unaware that it is being virtualized and requires no changes to work in this configuration.



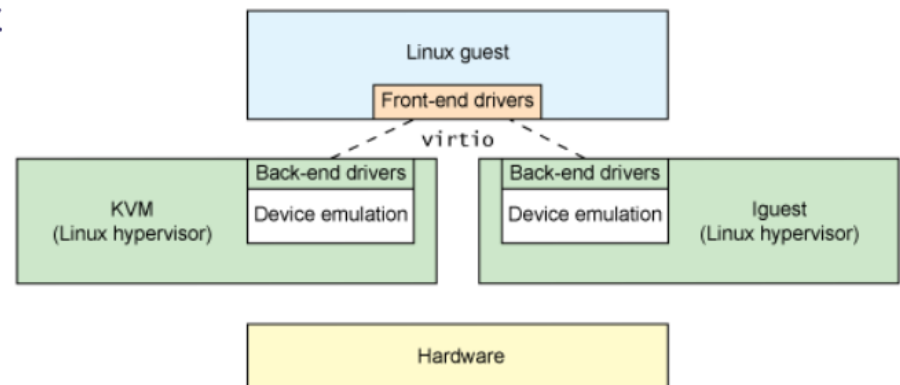
Conversely, in paravirtualization, the guest operating system is not only aware that it is running on a hypervisor but includes code to make guest-to-hypervisor transitions more efficient

virtio provides an abstraction of a set of device drivers commonly emulated by paravirtualized hypervisors

Goal: increase the reuse of code across platforms

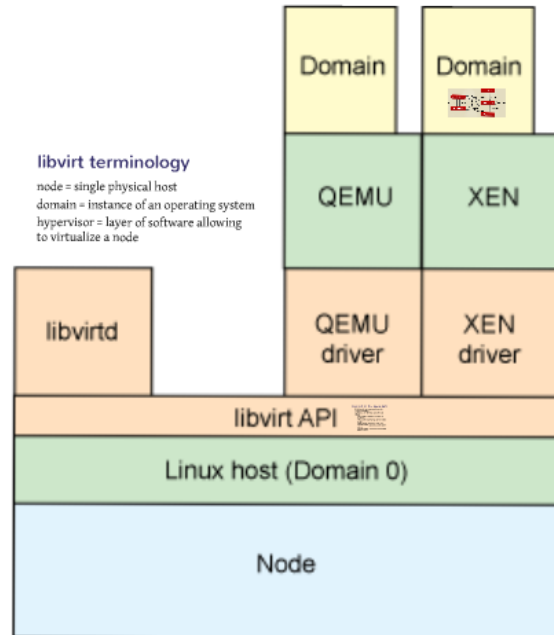
disk and network interfaces of the guest need to implement front-end drivers

Back-end drivers have not to be implemented in a common way, as long as they provide functions needed by the front-end driver



The libvirt API

With applications build on libvirt, it is possible to use multiple hypervisors on a host - also via network



libvirt uses a driver-based architecture:

- a large number of hypervisors are serviced in a common fashion
- special functions of certain hypervisors may not be visible in an API
- also, hypervisors have not to implement all functions of libvirt

libvirt terminology

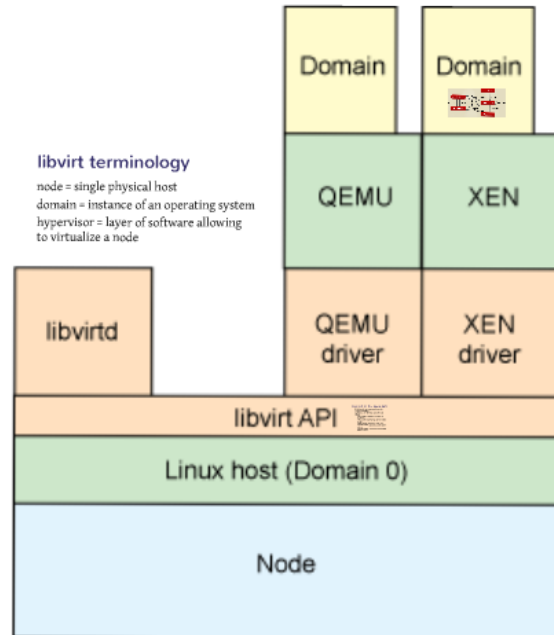
node = single physical host

domain = instance of an operating system

hypervisor = layer of software allowing
to virtualize a node

The libvirt API

With applications build on libvirt, it is possible to use multiple hypervisors on a host - also via network



libvirt uses a driver-based architecture:

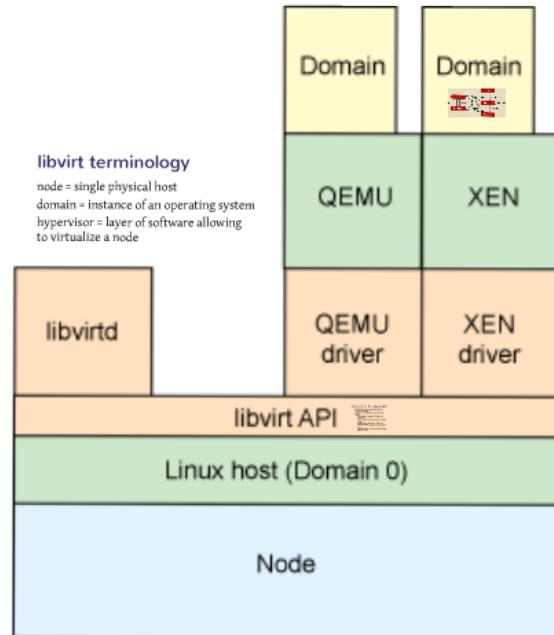
- a large number of hypervisors are serviced in a common fashion
- special functions of certain hypervisors may not be visible in an API
- also, hypervisors have not to implement all functions of libvirt

zoomed in: the libvirt API

- libvirt needs to expose resources in order to manage virtualization
- applications using libvirt can access five main objects:
 - virConnectPtr - represents connection to hypervisor
 - virDomainPtr - represents an active or defined domain
 - virNetworkPtr - represents one network
 - virStorageVolPtr - represents a single storage volume
 - virStoragePoolPtr - represents a pool, a logical area for volumes

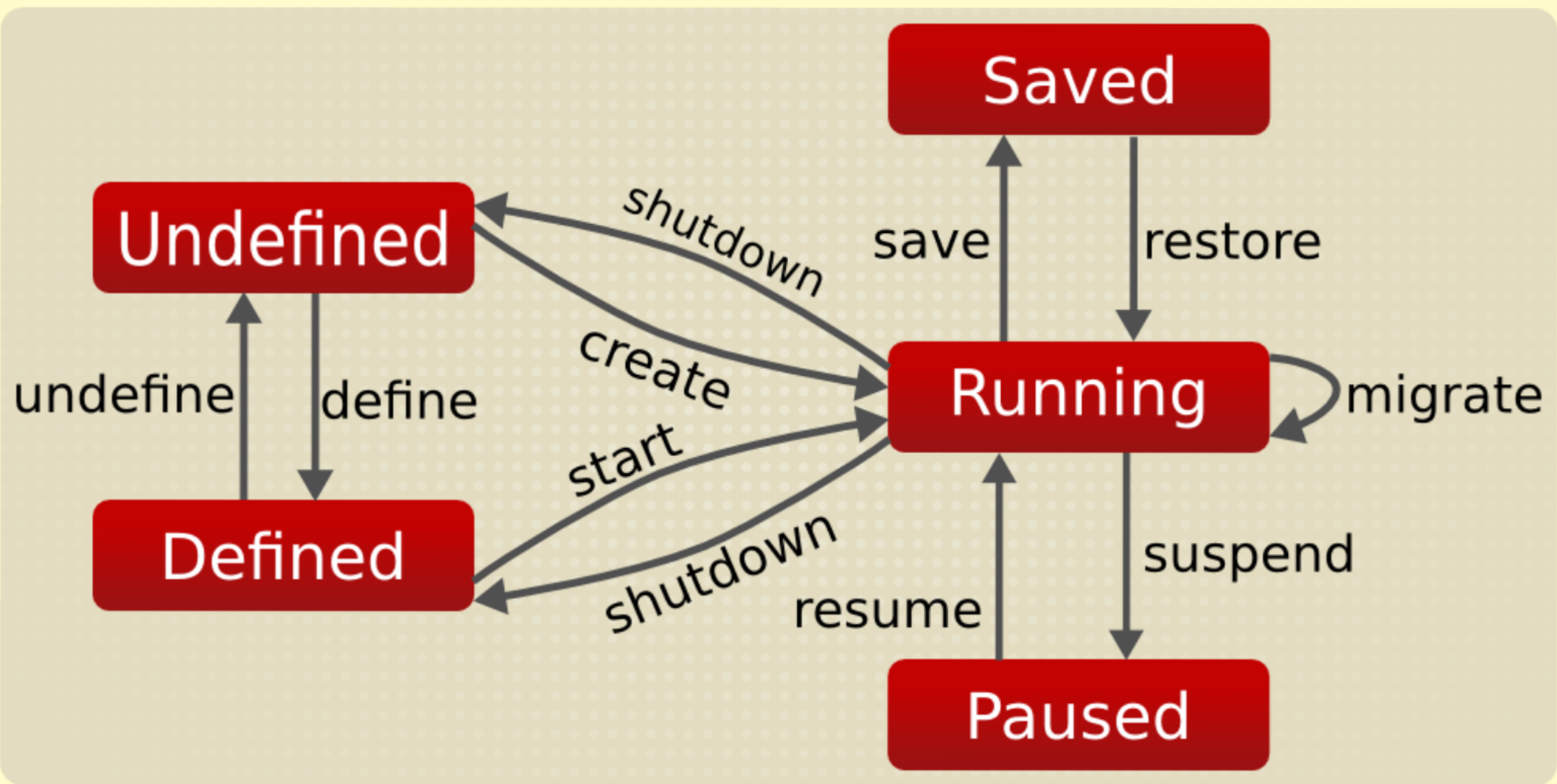
The libvirt API

With applications build on libvirt, it is possible to use multiple hypervisors on a host - also via network



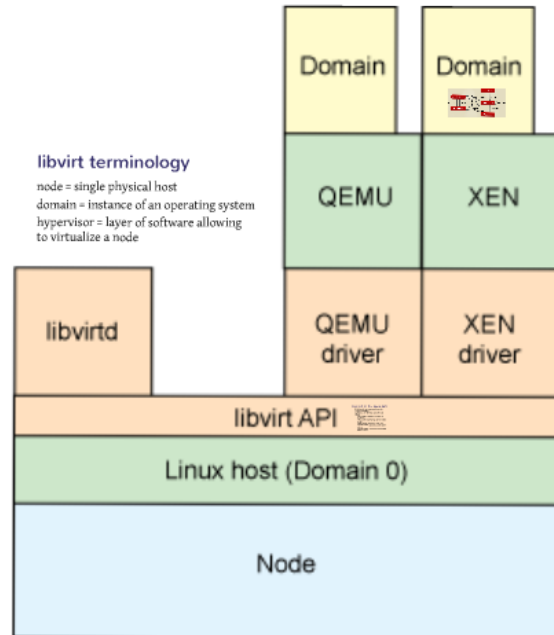
libvirt uses a driver-based architecture:

- a large number of hypervisors are serviced in a common fashion
- special functions of certain hypervisors may not be visible in an API
- also, hypervisors have not to implement all functions of libvirt



The libvirt API

With applications build on libvirt, it is possible to use multiple hypervisors on a host - also via network



libvirt uses a driver-based architecture:

- a large number of hypervisors are serviced in a common fashion
- special functions of certain hypervisors may not be visible in an API
- also, hypervisors have not to implement all functions of libvirt

libvirt applied

domain configuration uses xml-files:

- root element is named `<domain>`
 - attributes: type and id, they specify the hypervisor
- cpu configuration is given in `<vcpu>`
- memory is configured in `<memory>`
- network is added as a `<device>`
- see <http://libvirt.org/formatdomain.html>

Example is found here

```
<?xml version='1.0' type='text'>
<domain type='kvm' id='1'>
  <name>example</name>
  <os type='linux'>
    <type>hvm</type>
    <loader type='bios'>bios.bin</loader>
    <initrd type='initrd'>initrd.img</initrd>
    <kernel type='kernel'>kernel.img</kernel>
  </os>
  <cpu type='pentium'></cpu>
  <memory type='memory'>
    <unit type='KiB' value='1024'></unit>
  </memory>
  <disk type='file'>
    <source type='file' file='/var/lib/libvirt/images/example.img'></source>
    <target type='drive' device='hda'></target>
    <cache type='writeback'></cache>
  </disk>
  <interface type='network'>
    <source type='network' name='default'></source>
    <target type='bridge' device='vnet0'></target>
  </interface>
  <watchdog type='i6300esb'></watchdog>
  <devices>
    <device type='usb' target='usb-lcd'></device>
    <device type='usb' target='usb-mouse'></device>
    <device type='usb' target='usb-tablet'></device>
    <device type='usb' target='usb-tablet'></device>
    <device type='usb' target='usb-tablet'></device>
    <device type='usb' target='usb-tablet'></device>
  </devices>
</domain>
```

- root element is named `<domain>`
 - attributes: type and id, they specify the hypervisor
- cpu configuration is given in `<vcpu>`
- memory is configured in `<memory>`
- network is added as a `<device>`
- see <http://libvirt.org/formatdomain.html>

-

[illegible]

```
<domain type='qemu'>
  <name>my-L2-domain</name>
  <uuid>c7a5fdbd-cdaf-9455-926a</uuid>
  <memory>1536</memory>
  <currentMemory>1536</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='i686'
machine='pc'>hvm</type>
    <boot dev='cdrom' />
  </os>
  <devices>
    <emulator>/usr/bin/qemu-system-
x86_64</emulator>
    <disk type='file' device='cdrom'>
      <source file='/home/user/boot.iso' />
      <target dev='hdc' />
      <readonly />
    </disk>
    <disk type='file' device='disk'>
      <source file='/home/user/debian.img' />
      <target dev='hda' />
    </disk>
    <graphics type='vnc' port='-1' />
    <interface type='bridge'>
      <source bridge='br0' />
      <mac address="00:11:22:33:44:55" />
    </interface>
  </devices>
</domain>
```



VM migration

Migration of a virtual machine means moving it from one host to another

Requirements:

- Shared storage accessible under same paths and locations, e.g. iSCSI, NFS, etc.
- Exact same version of hypervisors on both hosts
- Same network configuration
- Same CPUs, or better said CPUs have to be from the same vendor and CPU flags on the destination host must be superset of CPU flags on source



x86 hardware virtualization

A prezi by Jens Preußner
supervised by Laszlo Kajan
May 8, 2012



or virtualize?

costs), and management costs
Less hardware means improved reliability

ulation of a system focuses on exact reproduction
of its behavior
primary goal of virtualization is to provide an abstract
which separates users from physical resources

machine) is a kernel
6 hardware
List of supported guest systems:
• Linux (32 and 64 bit)
• Windows (32 and 64 bit)
• Solaris
• AIX
• FreeBSD
• Solaris
• other BSD-Derivates

instruction
sor instructions

Advantages
Disadvantages
not dynamic, but uses
sparse file
no compression/
encryption
quicker than qcow
dynamic
VMware format
distributable,
can be mirrored

ins moving it from
path and location, e.g. SCSI, NFS, etc.
on both hosts
not be exposed of CPU flags on source

All in all ...

Platform virtualization brings not only technical advantages but cost and energy advantages, as well. This sounds fine nowadays.

Why virtualization?

Studies show: Only 5% of server capacity is used, the rest of the time, a server more or less "sleeps"!

Virtualizing platforms on a single server can improve server utilization:

- reducing server count
- reduced real estate, power consumption, cooling (less energy costs), and management costs
- Less hardware means improved reliability

All in all ...

Platform virtualization brings not only technical advantages but cost and energy advantages, as well. This sounds fine nowadays.



Thank you for your attention!

